Social Dependence Relationships in Requirements Engineering[⋆]

John Mylopoulos[1], Daniel Amyot[1], Luigi Logrippo[1,2], Alireza Parvizimosaed[1], and Sepehr Sharifi[1]

[1] School of EECS, University of Ottawa, Ottawa, Canada
{jmylopou, damyot, logrippo, aparv007, sshar190}@uottawa.ca
[2] Université du Québec en Outaouais, Gatineau, Canada

Abstract. *i** is a requirements modelling language that is founded on social concepts (actors, social dependencies). Azzurra is a business process specification language that uses roles and commitments as primitives. Symboleo is a smart contract specification language grounded in legal concepts (roles, parties, obligations, powers). We compare and contrast the social dependence relationships used by the three languages.

Keywords: Requirements Engineering · Social Dependency · *i** model.

1 Introduction

One of the key elements of *i** [3] is that of social dependency[3] that holds between two actors, a *depender* who depends on a *dependee* to satisfy a *dependum* (goal, task, resource, softgoal). This is a very powerful concept that constitutes the foundation for social modelling and has spawned interesting dependence relationships for software, business processes, and legal contracts. The purpose of this paper is to discuss the ontological nature and contrast three types of social dependence relationships: social dependencies (*i**), commitments (Azzurra) [2], as well as obligations and powers (Symboleo) [4]. These relationships were developed over three decades and involved many collaborators beyond the authors.

Given that the languages target different domains, different examples are used for illustration. Note also that a full comparison of these languages is beyond the scope of this paper.

2 Social Dependencies (*i**)

In an *i** social dependency (Fig. 1), the *depender* wants something and the *dependee* is able and willing to deliver it. However, the force of the dependence can

[⋆] Partially funded by an NSERC Strategic Partnership Grant titled "Middleware Framework and Programming Infrastructure for IoT Services" and by SSHRC's Partnership Grant "Autonomy Through Cyberjustice Technologies"

[3] We use the term *'social dependence relationship'* synonymously with *'social dependency'*.

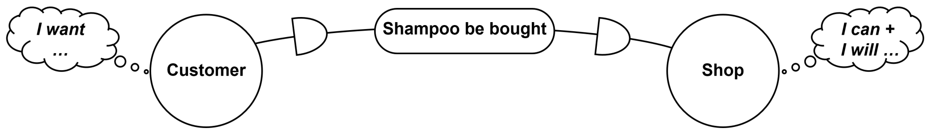

Fig. 1: Example of social dependence relationship in *i**

vary from weak to strong on either side of the *dependum*. Consider a customer's dependence on a shop to find her favourite hair shampoo: it is weak because there are probably other stores that sell it as well, and other shampoos are also comparable. Contrast this with one's dependence on a renowned surgeon for a rare medical operation. This one is strong, because there are no substitute dependees. The force of the dependence on the *dependee*'s side is even more important, as she is responsible for delivering on the dependence. That force is defined along two dimensions: ability to deliver on the *dependum* and degree of commitment to deliver. The dependence on the renowned surgeon is strong on ability and medium on commitment, as surgeons will postpone operations in case of an emergency.

Now, consider a beggar who depends on passersby to fulfill her goal of having some money. This is a social dependence too and it is weak on both sides: the beggar can switch to another kind of dependence to get some money, e.g, work, while the passersby have not even agreed explicitly to the beggar to deliver on the *dependum*, they are just willing to do it, occasionally. Here, the dependence is established statistically: some passersby give money, and sooner or later this establishes a dependency for the beggar.

*i** does recognize the importance of the force of a dependence by allowing three possible levels of force: critical, committed, and open [3]. But, as discussed in the sequel, it turns out that in other areas of research, people have opted for defining specializations of social dependence relationships where the strength of dependence is built into their semantics. Commitments, obligations, and powers are three such relationships.

*i** was developed at the University of Toronto in the early '90s as part of Eric Yu's PhD thesis, with collaborators Lawrence Chung, Brian Nixon, and John Mylopoulos. It further led to the development of many other languages and dialects, including the *Goal-oriented Requirement Language*, part of the User Requirements Notation standard, with collaborators documented in [1].

3 Commitments (Azzurra)

Originating in the area of multi-agent systems [5], commitments capture a social dependence where there is an explicit speech act executed "Agent A: I want X - Agent B: I commit to fulfill X". This kind of dependency has substantially more force than the beggar's dependence on passersby. It means that the *dependee* intends to deliver, provided some conditions hold. So, commitments are social dependencies that always arise from intentions, rather than mere practice, and are established through verbal acts. More formally, commitments are 4-tuples `C(debtor, creditor, antecedent, consequent)`, where the creditor is the *depender*, i.e., the beneficiary of the commitment being fulfilled, while the debtor is the *dependee*. Moreover, the commitment is fulfilled when the consequent becomes true, provided the antecedent is true. Moreover, commitments go through states, such as 'created', 'active', 'suspended', 'success', and 'failure'. Allowable state transitions are defined by a state diagram. Note that commitments constitute specializations of social dependencies, with better fleshed out semantics, proposed for use in multi-agent systems. They also come with a precise level of force for the debtor who intends to fulfill what she committed to, while the creditor has the right to expect that the commitment will be fulfilled. The passersby mentioned earlier have no commitment towards the beggar and, in turn, she has no right to expect anything from them.

Azzurra is a conceptual modelling language for business processes. Its main thesis is that such processes, being social artifacts, need to be defined in social terms, rather than system-oriented ones (e.g., Petri nets or BPMN). Accordingly, business processes (aka 'protocols') are defined in terms of roles and commitments, with constraints attached. Azzurra models can be seen as requirements specifications for business processes, as they describe *what* a business process is supposed to achieve without getting into the details of *how* to achieve it.

Figure 2 (adapted from [2]) presents an Azzurra protocol for fracture treatment. The protocol includes as parameters a hospital number that serves as key for treatment instances, a patient, and a specialist. It also includes role parameters, such as a radiologist and a surgeon.

There are nine commitments for this protocol, each using a <*trigger*> $\twoheadrightarrow$ <*commitment*> format. The first, C_1, is triggered when the protocol is instantiated, has as roles the specialist (debtor) and the patient (creditor), is unconditional (antecedent= true), and is fulfilled when the patient is i) examined, then ii) diagnosed, and then iii) dis-hospitalized. The second commitment, C_2, is triggered if there is no need for X-rays and is fulfilled when a sling is made for the patient. Protocol refinements constrain the agents that participate in a protocol instance. For example, agents may be constrained on how many concurrent commitments they have for a given role, such as a surgeon for treatment protocol instances. Finally, a knowledge base defines some domain axioms, which can be used to reason about propositions used as triggers or as elements of commitment antecedents and consequents.

Azzurra supports two types of reasoning. `ENACTPROTOCOL` determines how an event updates the state of a protocol instance and of the commitment instances therein. `CHECKCOMPLIANCE` checks whether an occurred event violates the specification of a protocol instance. This corresponds to identifying commitments that are not created/fulfilled, unexpected commitment operations, and protocol constraint violations.

In summary, commitments specialize and improve the formalization of social dependence relationships. They also come with a richer language than *i** for

protocol Treatment (**key** hospnr, pt : Patient, sp : Specialist) {
ag-variables: rc : RehabCentre, ra : Radiologist, or : Orthopedist,
su : Surgeon, nu : Nurse;
commitments:
init $\twoheadrightarrow$ C_1 : C(sp, pt, $\top$, Examined $\cdot$ Diagnosed $\cdot$ Dehospd) final
NoXRayNeeded $\twoheadrightarrow$ C_2 : C(or, sp, $\top$, SlingMade)
XRayRequested $\twoheadrightarrow$ C_3 : C(ra, sp, $\top$, XRayPerformed)
XRayRequested $\twoheadrightarrow$
C_4 : C^*(sp, ra, XRayPerformed, FractAssessed)
FractAssessed $\twoheadrightarrow$ C_5 : C(or, sp, $\top$, ((Fixated$\oplus$Plastered)
$\vee$ fulfil(C_6) $\vee$ SlingMade))
FractAssessed $\twoheadrightarrow_{\leq 2h}$ C_6 : C^*(su, or, SurgeryRequested,
Operated)
Operated [$\neg$fused] $\twoheadrightarrow$ C_7 : C(nu, pt, $\top$, RcChosen(rc))
RcChosen(rc) $\twoheadrightarrow$ C_8 : C(rc, pt, $\top$, fulfil-p(RehabGiven,
key=hospnr, pat-id=pt, ref-sp=sp))
MedPrescribed(m) $\twoheadrightarrow$ C_9 : C(nu, sp, $\top$, MedApplied(m))
can-deleg-no-resp(C_3)
deadline(C_2, 2h)
protocol refinements:
role-confl(Radiologist,Orthopedist)
kb:
implies(XRayRequested, Diagnosed)
implies(NoXRayNeeded, Diagnosed)
implies(MedPrescribed(m), Diagnosed)
mutExcl(XRayRequested, NoXRayNeeded) }

Fig. 2: An Azzurra protocol for fracture treatment (from [2])

modeling business processes in an outcome-oriented way. Azzurra was developed at the University of Trento between 2012 and 2016 and only managed a few publications including a best paper, awarded at RCIS 2015. The collaborators for that project included Fabiano Dalpiaz, Evellin Cardozo, Giulia Canobbio, Paolo Giorgini, and John Mylopoulos.

4 Obligations and Powers (Symboleo)

Obligations are commitments with legal force. The legal force is defined through *powers* that a creditor has towards the debtor of an obligation to cancel or suspend an obligation or another power, or to initiate new obligations or powers. Obligations constitute a specialization of the concept of commitment in that they can be created, cancelled, etc. by someone who has the power to do so. In turn, powers constitute specializations of obligations in that they can include in their consequent the creation, cancellation, etc. of other powers or obligations.

Obligations and powers can be found in *legal contracts*. Legal contracts can be thought as processes too, but they are legal rather than business ones, in that they describe the space of allowable executions that comply with terms and conditions of a contract. The presence of powers in legal contracts make them a much more malleable concept than that of a business process in that they can be reshaped through powers with the introduction/cancellation of obligations while a contract is being executed (i.e., “performed” in Law).

Symboleo is a formal specification language for legal contracts, intended to formalize requirements for smart contracts. These are software systems running

Table 1: Abbreviated Symboleo specification for a goods sale contract

```
Domain salesD

  Goods isA Asset with goodsID: Integer;
  ...
  Delivered isA Event with delAddress: String, delDueDate: Date;

endDomain
Contract salesC(seller: Seller, buyer: Buyer, ID: Integer, amnt: Integer, curr: Currency, de-
lAdd, delDd: String)

  Declarations
    /* Values of parameters are passed on to the variables defined in the domain model. */
    goods : Goods with goodsID := ID;
    ...
    delivered : Delivered with delAddress := delAdd, delDueDate := delDd;
  Preconditions
    isOwner(seller, goods) AND NOT isOwner(buyer, goods);
  Postconditions
    isOwner(buyer, goods) AND NOT isOwner(seller, goods);
  Obligations
    O1 : O(Seller, Buyer, true, happensBefore(delivered, delivered.delDueD));
    O2 : O(Buyer, Seller, true, happensBefore(paid, paid.payDueD));
  Powers
    P1 : violates(O2, _) → P(Seller, Buyer, true, terminates(salesC));
  SurvivingObl
    /* Some obligations may remain active, e.g., confidentiality obligations. */
  Constraints
    not(isEqual(buyer, seller));

endContract
```

on a blockchain platform (or involving a regular database) that monitor and control the execution of a legal contract. Symboleo is founded on an ontology for contracts that is centered around the notions of obligation and power, and includes role and party (the actors playing roles in a contract), asset, as well as situation and event. Situations occur over time, e.g., the situation of commuting to work. Events, on the other hand, happen instantaneously, e.g., *arrivedAtWork*.

The language adopts many elements from Azzurra. Obligations and powers use the same format as commitments. Their antecedents, consequents, and triggers are expressed as events happening in a certain order and satisfying constraints. For example, for a sale contract, the consequent of a delivery obligation may be "Sale item delivered to delivery address by delivery date".

Table 1 presents a Symboleo specification for a contract (adapted from [4]). As shown, a Symboleo specification begins with the description of concepts in the domain. These are defined as classes that specialize concepts in the Symboleo ontology. For example, `Goods` specializes `Asset` and has an additional attribute `goodsID`. Instances of this class include sale items involved in a sale transaction. The domain model for a contract is followed by declarations of variables that take as values instances of domain classes. Pre/post-conditions have the same semantics as in program specifications.

The core of a contract specification are its obligations and powers. In our example there are two obligations: the seller must deliver the sale item to the delivery address by the delivery date (O_1), while the buyer must pay on time the sale amount (O_2). The contract also includes one power: if the buyer violates the payment obligation, the seller has the power to terminate the contract (P_1). Note that the creditor of a power may choose to not exercise it.

Legal contracts are complex constructs with many features that go well beyond those of business processes. For instance, some obligations may apply after the successful termination of a contract (and are accordingly called *surviving* obligations), e.g., a confidentiality obligation for a sale transaction may apply 6 months after a contract terminates. Contracts may spawn subcontracts that may be established while a contract is executing. For example, when a developer undertakes a large project in the construction industry, she may not have lined up all the subcontractors and their respective subcontracts.

Symboleo specifications can be validated to ensure that they are consistent with the expectations of the contracting parties by a tool (https://doi.org/10.5281/zenodo.3903954) that enacts scenarios and determines the contract final state. For example, for the scenario "Seller delivers on time, buyer does not pay on time, seller exercises power to terminate", the tool determines that the final state of the contract is 'cancelled'. The scenarios for validation are provided by contracting parties, along with their anticipated final state of the contract when each scenario is enacted.

The Symboleo project started in January 2018 and is ongoing at the University of Ottawa with the co-authors of this paper as main contributors.

5 Conclusions

The concept of social dependence relationships in RE was pioneered by *i**. This provided a solid conceptual base that is being proven to be of lasting value for developments in other application domains. With changes in syntax and semantics, it led to formalizing the concept of commitment in business processes in Azzurra. Then came the introduction of contractual obligations and powers in Symboleo, which leads to the possibility of monitoring legal compliance in contracts, possibly with blockchain support. We expect the social dependence concept to influence other languages in the future.

References

1. Amyot, D., Mussbacher, G.: User Requirements Notation: the first ten years, the next ten years. Journal of Software (JSW) **6**(5), 747–768 (2011)
2. Dalpiaz, F., Cardoso, E., Canobbio, G., Giorgini, P., Mylopoulos, J.: Social specifications of business processes with Azzurra. In: 9th RCIS. pp. 7–18. IEEE (2015)
3. Eric, S., Giorgini, P., Maiden, N., Mylopoulos, J.: Social modeling for requirements engineering. MIT Press (2011)
4. Sharifi, S., Parvizimosaed, A., Amyot, D., Logrippo, L., Mylopoulos, J.: Symboleo: A specification language for smart contracts. In: 28th IEEE International Requirements Engineering Conference (RE'20). IEEE CS (2020), *(to appear)*
5. Singh, M.P.: An ontology for commitments in multiagent systems. Artificial intelligence and law **7**(1), 97–113 (1999)